Software Development Practices

CICE Consortium

CICE Consortium Member representatives and community developers should strive to incorporate these practices throughout the development process, to the extent possible:

Process

- Define tasks of team members ahead of time, including coordination roles
- Utilize a common communication space for chats and discussions
- Work in shared repositories
- Adhere to the documentation strategy
    - Provide concise comments throughout the code
    - Write readable guides for build/run and a scientific overview
    - Deposit namelist settings, confidence scores and other documentation for tested and validated configurations in the repository
    - Document and track requirements and issues
        - Describe code changes and their effects on performance and physical result
    - Document “lessons learned” e.g. configuration errors, compiler errors, software versions and dependencies, FAQ, etc
- Manage code dependencies
- Practice Test-Driven Development
    - Develop tests first and upgrade them as the software development evolves
    - Automate as much as possible
    - Run standard testing suite regularly
- Practice version control on shared branches
    - Check in only working code (i.e. does not crash or give unexplainable results)
    - Check in only necessary code changes in minimal chunks (one “fix” at a time)
    - Maintain unedited, reproducible tagged and released versions
- Practice continuous integration
    - Frequent check-ins to the common repository
    - Frequent updates of private development branches from public branches
    - Automated testing
- Provide frequent technical and scientific reviews of
    - Automated test suite
    - Source code
    - Documentation
    - New features implemented with namelist (off by default)

    - Bit-for-bit expectations
    - Test summary
    - Changes needed in test coverage
- Tag and release code frequently

Coding Standard

- Choose a simple design
- Adhere to the current coding and naming conventions
    - See the CICE module ice_state.F90 for category conventions
    - See table below for file naming conventions
- Write readable code
    - Use meaningful variable names
    - Indent 2 or 3 spaces for loops, conditionals, etc
    - Vertically align similar elements where it makes sense
    - Provide concise comments throughout the code
- Write efficient code
    - Use intrinsic functions or mathematical constructs instead of conditional statements, where possible
    - Monitor performance changes throughout development
- Make interfaces modular
    - Pass most variables and constants through subroutine arguments instead of use statements
    - Declare most constants in shared modules, do not hard-wire constants
- Maintain bit-for-bit output from the default configuration (to the extent possible) via namelist flags
- Maintain global conservation of heat, water, salt

CICE and Icepack file naming convention	
CICEcore modules	ice_*.F90
CICEcore namelist	ice_in
Icepack driver modules	icepack_drv*.F90
Icepack namelist	icepack_in
Icepack interface modules	icepack_intfc*.F90
Icepack modules	icepack_*.F90

Document history

16 January 2017: Initial draft
1 July 2017: Dated for public posting